DLW Security Track: Documentation

Public GitHub Repository: https://github.com/dan-svasti/dlw-track-3

System Overview and Link to Problem Statement

This solution is a system that flags video footage with unusually large crowds in rural areas that are supposed to be sparsely-populated, as well as groups that loiter for an extended period of time.

In relation to the problem statement, this aims to address the gap in 'early detection'. While not an 'incident forecasting' tool, the system adds spatial awareness by warning the end user (authorities) of potential risks to rural safety. Moreover, it will capture 'safety-relevant signals' (crowds and loiterers) to "support decision-making" of authorities, not replace their judgment. Focusing on crowds in rural, low-population areas specifically is relevant to the context, as this is one of the few situations where a large group of people in one area would be an inherent anomaly. One may ask, "Why does that street corner have 3 people standing around for several minutes at night?" or "Why does that bus stop with an average of 2 people now have 12 people waiting?", which aren't reasonable complaints for densely-populated areas, but understandable for sparsely-populated ones. Furthermore, we attempt to respect privacy as the technology is designed to be integrated into existing CCTV infrastructure (or new ones in public areas). Lastly, considering rural technological constraints, we will attempt to make the solution lightweight, where all processing takes place locally at the camera, and only a singular alert signal is sent to the user.

The goal is to accept a video of a certain setting as input - much like CCTV footage focused on one area - compute a rolling average of the number of people at said setting over time, generate a certain threshold for an anomalously high number of people, and output an alert message if said threshold is crossed for a certain amount of time. This alert message will contain the estimated number of people spotted, the baseline average, the timestamp at which this was reported, and the amount of time for which they have remained in the area. A frame with bounding boxes around suspicious people will also be outputted as evidence to aid authorities' decision-making.

Libraries, Software, and Datasets

Due to time constraints and other reasons below, we will be leveraging the YOLOv8 Human Detection Model **(pre-trained)** and transfer learning to create the model. More specifically, the nano version will be used as it is the most lightweight solution, which is needed for this context,

even if we sacrifice computational power. OpenCV will be the library used for video-processing functionality, which supports pre-trained models.

One thing to note is that while it is preferred for a hackathon solution to have a model trained by us, we have to consider the context of the task. For rural safety, it would be much more reassuring to authorities to use a tried-and-tested deep learning model rather than a newly-trained one, especially considering the possible loss of human life if suspicious activity is not flagged.

As for the data used, test video clips were hand-picked from a CCTV anomaly detection dataset on Kaggle [1] (this included both normal and crime footage) to ensure that they had at least a suburban environment (rural environments are quite rare on public datasets).

Planned System Flowchart: Basic Functionality

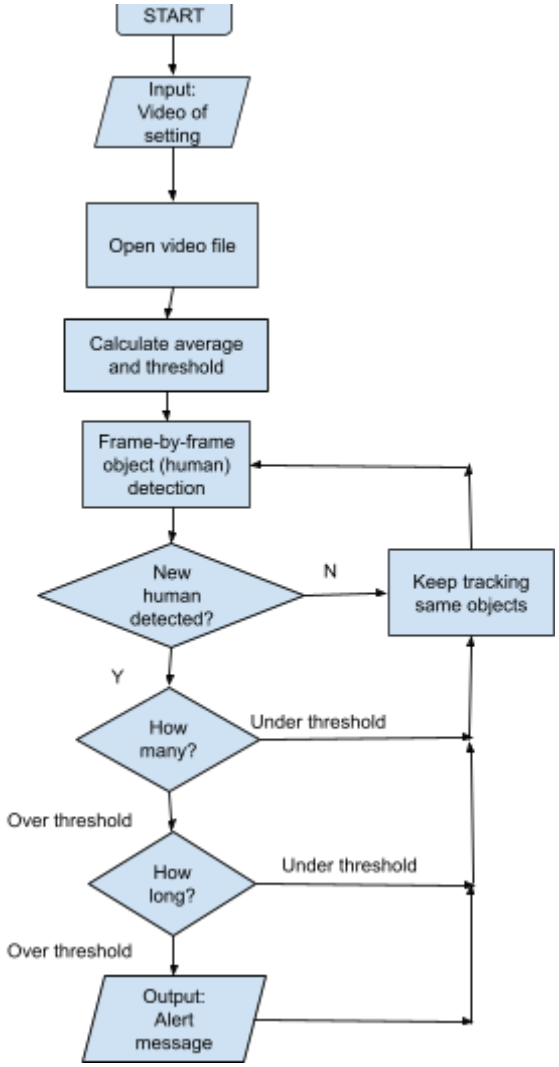

Fig 1: Draft of basic system flowchart

Key Choices in Developing the Product

Counting People Per Frame

When using YOLOv8 to compute the number of people over every frame, we decided that a confidence threshold of 0.5 was ideal as tests with 0.4-0.49 had prolonged periods (1-2 secs) of mistakenly identifying inanimate objects as humans, as opposed to split-seconds with a higher interval, although we also had to balance it with the possibility that a human was not detected (some humans had a ~0.6 confidence for a few frames). Moreover, using “stream = True” allowed for frame-by-frame, real-time processing, which is more memory-efficient, by way of returning a generator object rather than loading all frames into RAM [2].

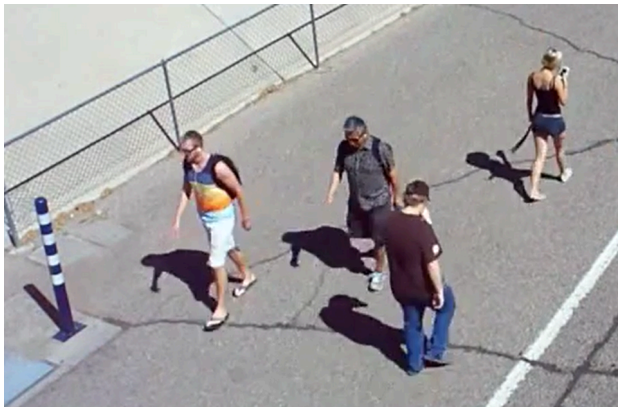

Fig 2.1: Test, 30-sec video of pedestrians walking on a suburban road

Fig 2.2: YOLOv8 human detection on Fig 2.1

Using Statistics to Detect Anomalies

Rather than computing rolling averages that adapt by the second, which may consume more resources than desired for a low-resource, rural environment, we initially chose to prioritize realism by ‘reading’ the video before anomaly detection and finding the overall mean number of

people and standard deviation. This was because it is very likely that authorities had an idea of the average number of people in an area, also accounting for the fact that they could analyze past CCTV footage to ascertain this information.

However, we realized that because the system's goal is to *specifically* detect prolonged spikes in crowd size, such a crowd while 'reading' would skew the mean in such a way that it becomes less sensitive to any drastic increases during anomaly detection. Thus, we decided that the median and median absolute derivation (MAD) was better in this context due to robustness when faced with the possibility of outliers. Specifically, we used "crowd size > median + K * MAD" as the outlier condition, deciding to set K = 3 as, assuming normal distribution equivalence, would only contain 2.5% of the data at random.

	frame	people	time_sec	is_crowd_anomaly
296	296	6	12.333333	True
312	312	6	13.000000	True
314	314	7	13.083333	True
315	315	6	13.125000	True
316	316	6	13.166667	True

Fig 3: Demonstration of frames with outlier-sized crowds (realistically, there should have been none – look at the time frame)

Using Time to Consolidate Anomaly Detection

Following the completion of using statistics to detect anomalies, there is the obvious issue of indiscriminately reporting crowds of outlier size regardless of time spent in frame, i.e. loitering, which can lead to crime. Therefore, we included some code blocks that clearly define that a time threshold must be crossed to qualify as behavior worth reporting. We set a conservative threshold to 10 seconds to give time for people to move out of frame; while seemingly short, the dataset's footage all covered relatively small areas. As for smaller groups that were not outliers, set to a minimal threshold of 2 people, we gave them triple that time as people simply waiting, coincidentally within a frame, cannot immediately be associated with suspicious activity in the same manner as an assembly of a large crowd. Of course, in many rural environments, it is worth noting that even two people can be considered a crowd due to the sheer density of zero-person frames (much like most rural settings), so all loitering cases would be shown as crowds.

Alert System

The goal here was to output the findings of the previous sections in such a way that was human-readable, and with appropriate evidence. We used OpenCV's built-in functionality to lock on specific frames, read them, and save them to an "evidence" folder for the end user.

```python
def format_alert(event, location_name="Camera 1"):
    return (
        f"ALERT: {event['type'].upper()} DETECTED\n"
        f"Location: {location_name}\n"
        f"Start Time: {event['start_time']:.1f}s\n"
        f"Duration: {event['duration']:.1f}s\n"
        f"Max People Detected: {event['max_people']}\n"
    ) # alert message function format


for event in events: # prints an error message for every incident,
    alert_msg = format_alert(event)
    print(alert_msg)
    print("-" * 40)
```

```
ALERT: CROWD DETECTED
Location: Camera 1
Start Time: 0.0s
Duration: 28.1s
Max People Detected: 4

----------------------------------------
```

Fig 4.1: This particular 6-min video had two crowd alerts; this is one example.

```
!ls evidence # sanity check
```

```
crowd_422.jpg  crowd_6448.jpg
```

Fig 4.2: Save image files as evidence

Fig 4.3: Example of evidence frame saved to “evidence” folder

Suggestions for Further Improvement

While we chose for the model to accept recorded video clips as input, this model would not be of significant applicability in rural security until it runs in real-time on live footage. If we are working with live videos, then the outputted timestamps and frames have to be replaced with more relevant details such as the current time and the camera number/location, if the authorities

are monitoring multiple cameras at multiple locations. Moreover, regarding a possibility of increased sensitivity to suspicious behavior at night, it would be much more accurate to use the time of day rather than judging by the environmental conditions, such as brightness.

Furthermore, one disadvantage with using predefined medians for the recording is the lack of flexibility for certain events, such as seasonal celebrations, children returning home from school, etc. This could be solved with rolling averages (computed in parallel with anomaly detection), although one could justify fixed averages, arguing that such events involving a large volume of people should come to the attention of authorities to ensure the safety of everyone nearby.

Additionally, the system should have the functionality to keep track of each individual person - of course, this may require more computational power, but is certainly more useful for identifying individual loiterers rather than sustained counts of people.

Lastly, we know that loitering and large crowds are far from the only signs of imminent crime - the system does not account for aggressive body language, carrying of weapons, or a person(s) following someone for an extended period of time, to name a few more suspicious behaviors. That being said, it would be very beneficial to develop the solution into a full-fledged, CCTV-compatible system that utilizes deep learning to detect incidents before they occur. However, this will require additional datasets and training that YOLOv8 alone cannot accomplish, which may be less lightweight than rural authorities hope. Thus, extending this solution in particular requires the funding and setup of additional surveillance infrastructure in rural communities, which isn't always feasible.

References

[1] webadvisor, "Real Time Anomaly Detection in CCTV Surveillance," *Kaggle.com*, 2022. Available: https://www.kaggle.com/datasets/webadvisor/real-time-anomaly-detection-in-cctv-surveillance

[2] Ultralytics, "Model Prediction with Ultralytics YOLO," *Ultralytics.com*, 2023. Available: https://docs.ultralytics.com/modes/predict/#why-use-ultralytics-yolo-for-inference. [Accessed: Jan. 27, 2026]